Silicon Mountain Computers
C-12, Mtn. Stn. Group Box
Nelson, BC V1L 5P1
Canada

HIGH-MEMORY HOT-Z

INTRODUCTION

"HI-Z" is a modification of Ray Kingsley's HOT-Z II for the ZX81/TS1000/TS1500. I developed it primarily for my own uses as a software developer; however, Ray and I both felt that other users could benefit from this, so decided to make it available to licensed users of Hot Z at a nominal update charge.

HI-Z is completely relocated into the 8000-BFFF block of memory (32-48K). The beauty of this is that Hot Z becomes totally "programmer transparent," leaving the entire "user RAM" space from 4000-7FFF free for your applications. You can therefore quit Hot Z, load, run, and/or test your own programs, and jump right back into Hot Z to make changes or write new code, LEAVING YOUR PROGRAM COMPLETELY INTACT.

For example, I recently wrote a long machine-code game whose code and data takes up the entire region from 2000 to 8000h. Without HI-Z, such a program could have only been written by the "patchwork" method; time-consuming, frustrating, and highly prone to relocation errors. Using a conventional assembler, such a chore could take years for the average casual programmer. Even with the original Hot Z, it could take many months. With HI-Z, the task was greatly simplified, making it possible to write some 12K of code within about six weeks.

Since HI-Z initializes RAMTOP to 8000h and then places itself above this, it is also highly immune to what I term "soft" crashes; i.e. crashes that only reset the machine without rampant runaway LDIR's, etc.

Additional advantages will become clear as we cover the "fine points" of the HI-Z version of Hot Z.

HARDWARE REQUIREMENTS

THEORY

Most of you know that you can't run machine-code above the 32K mark. The reason may not be exactly clear, so here is a basic explanation. It has to do with the way the video display system was designed. Whenever address line A15 is high (32-64K being addressed) during an instruction fetch cycle (M1 NOT), the ULA (custom logic chip) swings into action and forces NOPs onto the CPU data bus. It in turn reads the "instructions" contained there as data for video display, relying on a high-memory echo of the display file. This data is then serially sent to the modulator to produce each horizontal TV line. The first such "instruction" with bit 6 high is taken as a signal to the ULA that the display cycle is over, causing it to return M1 control to the CPU. (So you see, it's no accident that the display-file end markers are HALT commands!) The bottom line is that only opcodes with bit 6 high will execute in high memory; the rest are simply read as NOPs! Pretty useless, alright.

Fortunately, only the top 16K is really needed for this. By including A14 in the ULA M1 NOT decoding, we can restore true CPU instruction fetches in the 8000-BFFF region. Note, however, that doing this prevents us from putting the display file there. Fortunately, there is little or no commercial software that relies on this "trick," so M1 NOT decoding only adversely affects the occasional large home-brew BASIC program.

PRACTISE

If the above discussion was over your head, don't worry about it. All you really need to know is that you need a minor modification to your computer to make HI-Z work with your 64K RAM pack. Details were published in SyncWare News, Vol. 2 #5. Those of you who dislike hardware hacking may be pleased to know that I'll make up a plug-in adaptor to do the job for $15.

LOADING

After powering up, simply LOAD "HI-Z" or LOAD "". When loading is complete, the good old familiar Hot Z cover screen will come up. As before, you can make a back-up save by pressing S, or press any other key to enter Hot Z.

But now you may also press the BREAK key. I completely re-worked Ray's loader, putting the whole works into BASIC. No longer is your uploader tucked away in the printer buffer, where it is instantly cleared the moment you return to BASIC and enter a command. The entire Hot Z code and data is stored in dimensioned string array A$. An ordinary old BASIC PRINT statement prints the cover screen, and an ordinary old INKEY$ waits for a keypress, etc. As a result, the HI-Z loader will be very easy to adapt to other mass-storage devices. Do note, however, that you must never use RUN, CLEAR, or redimension A$, or else the HI-Z code and data will be lost.

The actual uploader routine is stored in the traditional "1 REM" area (4082). After entering Hot Z, the loader program will still be in place, so you can look it over and explore at leisure. (Hot Z disassembled itself; HI-Z even disassembles where it came from!)

COLD AND WARM BOOT

As mentioned earlier, RAMTOP is initialized at 8000, with the stack immediately below. You can therefore QUIT to BASIC, NEW, and have your whole 4000-7FFF area available for loading programs. To start Hot Z from a "cold boot," enter RAND USR 38912 (or other variant, such as RUN USR 38912, IF USR 38912 THEN, etc.). This is exactly like USR 22528 in the 16K version, and fully initializes the user stack, names files, variables, and so on. A cold boot may be done from either FAST or SLOW mode.

When working on a program it is often advantageous to re-enter Hot Z using a "warm boot." This leaves your variables, user stack, and names files intact; it even returns you to the next disassembly screen after the one you quit from. This warm entry point is at name CH00 (USR 43657). Note, however, that warm boot must always be from SLOW mode. If you accidently use FAST mode, all is not lost; your screen will be grey, but the program is actually running. Press ENTER to be sure you're in disassembly mode, followed by SHIFT Q to return to BASIC. Then enter SLOW mode and re-invoke the warm boot USR call.

MEMORY MAP

It is worthy of note that EVERY element of Hot Z has been moved from its original location in the supplied versions. Doing partial relocations involving only one or two elements is relatively straightforward, as described in the Hot Z documentation. However, if you ever attempted a full Hot Z relocation, you probably found out that there's more than meets the eye. It takes a lot of trial and error, and no less than FOUR separate relocations plus a few manual touch-ups to get everything moved into high memory. The HI-Z memory map now looks as follows:

8000-80A7 HZ variables
80A8-81DC Jump tables
81DE-885F HZ files
8860-9951 Free space for names
9952-97FE Resident names
9800-BF1F HZ Program
BF20-BFFF Unused

The region from BF20-BFFF is a handy place to put some of your own little utilities such as code movers, checksum routines, and such.

EVEN MORE ON NAMES AND NAMING

You'll note that there's a great big gap, from 8860-9951, that is unused. I set it up this way to allow expansion of the resident names file. Again by way of example, the 24K program referred to earlier simply didn't have any room for its own alternate names file in low memory. What you can do in a similar situation is to use this blank area for the ALNA files. Even better yet, simply add to the resident names!

There are two things to take into account when adding your own names to the resident file. The first is that a cold boot will reinitialize NTOP to the original setting, i.e. 9552. So after you add your names, press SHIFT T from disassembly mode to get the current NTOP. Write this address to locations 8854 and 8856 in the files area (as usual, in low-high order). This will cause a cold boot (intentional or accidental, as by stumbling into an error trap or hitting SHIFT R by mistake) to respect your new NTOP.

The other consideration is that the expanded names file will no longer fit into the loader. Some concessions are always necessary, and I felt that the capability of loading into a machine with RAMTOP at 8000 (as on power-up) overweighs the disadvantage of not being able to expand the names file. You can get around this in one of two ways:

1: Make up a new loader, redimensioning A$ as required and modifying the uploader to suit your new block sizes.

2: Use alternate mass-storage to save the entire 8000-BFFF block. For example, the A&J Stringy-Floppy, the popular Z-XLR8 tape package, and at least two DOS packages allow such a code-save option. This essentially makes the loader completely un-necessary.

I've taken the liberty of adding some of the most common ROM calls to the names file, but didn't go "whole-hog" on naming ROM routines. Personally I

prefer to have plenty of space available for whatever application I happen to be working on.

OTHER POINTS

A minor bug that plagues Oliger video systems has been patched. This caused goofy behavior (even crashes) when trying to use the tape routines. Note, however, that when using the tape commands with Oliger video you will NOT get the border colour changes. Slightly annoying, but not fatal.

A cold boot will re-initialize RAMTOP to 8000h. So if you move RAMTOP for whatever reason, be sure to use the warm boot to re-enter Hot Z after quitting to BASIC. For example, you might want to put the stack at C000 or higher, so that it's completely out of the way of your "user" space from 2000 to 8000.

You might note that HI-Z writes FA00 to (RAMTOP). This is for the benefit of Compusa disk owners; this system uses these bytes to point at its directory. FA00 is about as high as you can put it without running into trouble.

BEWARE of overflowing the user stack, or you'll start running into the machine-stack! TROUBLE!

As with earlier versions of Hot Z, you should be careful when single-stepping routines that write to the "unused" system variables (MEM6, printer buffer, etc.). The single-stepper relies on these to simulate registers, program counter, etc. Mess with this while single-stepping.... CRASH!

The printer routines were patched so that they work with the Memotech or Aerco printer interfaces, as well as with the TS2040. However, I found that occasionally the PRINT SCREEN (read mode shift Y) will print a double screen. I'm not sure why this is, but I think it has to do with the END setting. Put END somewhere on the current screen, and this shouldn't happen.

SOME BUGS AND GLITCHES

Considering how many tries it took before this relocation worked at all, I was amazed that it ended up working as well as it does. To my knowledge, only one bug crept in along the way, and even this will be serendipitous to many users (myself included). For some obscure reason, the character codes are no longer printed after commands such as LD A,n and CP n. This will be a real boon to users of the Memotech interface, which can crash gloriously on some of the graphics codes. (For those of you who really really miss this feature, hang tight. Ray has agreed to look it over and figure out what went wrong.)

This next one is not so much a bug as it is an annoyance that plagues all versions of Hot Z, including HI-Z. After quitting to BASIC, the error-handling system is left slightly out of whack. Unless you have the program cursor early enough in a BASIC program to give a 5/0 report on quitting, the next command you enter will zoof itself if it exceeds the length of one line. It's annoying to type in some complex calculation, only to have it refuse to auto-scroll and give a 5/(garbage) report when you reach the end of the line. I've gotten into the habit of typing FAST FAST FAST FAST FAST FAST FAST (shift F seven times) followed by DELETE, whenever quitting results in a K

cursor instead of a 5/(garbage) error code. This restores the error-handler, and subsequent command lines will auto-scroll like they should.

If your application (or hardware) messes with location 407B (HZET), you might get goofy behavior or even crashes if you stumble into an error trap (RST 08) after a warm boot. Avoid this either by leaving 407B alone, or by avoiding RST 08!

Be careful about single-stepping IN and OUT commands, especially IN. I have had the single-stepper freeze up on occasion while single-stepping these commands. (The results you get are usually meaningless anyway, so you're best off just skipping I/O commands entirely.)

BEWARE of naming (shift G) or deleting names (shift 3) from data mode. At best it won't take, at worst you'll lock up.

EPROMing HI-Z

It might be really neat to have HI-Z on a 2764 EPROM, mapped at 8000-BFFF. I haven't done it myself, but it should be quite easy to do. The only thing you need to remember is that HI-Z's read/write variables are at 8000-80A7. You'll therefore have to move these into RAM somewhere, as in low memory (2000-3FFF) or above C000. Since this is a one-step relocation, it should pose no problems. See Ray's notes on relocating Hot Z.

Fred Nachbaur